UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA & CENTRO DE CIÊNCIAS EXATAS E DA TERRA
DEPARTAMENTO DE INFORMÁTICA E MATEMÁTICA APLICADA

Um Sistema Web para Monitoramento e Controle de Reservatórios de Petróleo

por
Hércules Lisboa de Aquino Sobrinho
(herculesaquino@hotmail.com)

Monografia submetida ao Departamento de Informática e Matemática Aplicada da Universidade Federal do Rio Grande do Norte para obtenção do título de Engenheiro de Computação. Área de Concentração: Sistemas de Informação

Trabalho orientado por
Thaís Vasconcelos Batista
Professora Adjunta do Departamento de Informática e Matemática Aplicada da UFRN

Natal — RN, Julho de 2005.

Relatório de Graduação sob o título “Um Sistema Web para Monitoramento e Controle de Reservatórios de Petróleo”, defendido por Hércules Lisboa de Aquino Sobrinho em 20 de julho de 2005, em Natal, Rio Grande do Norte, pela banca examinadora constituída pelos professores:

____

Prof. MSc. Adilson Barboza Lopes
Departamento de Informática e Matemática Aplicada
UFRN

____

Prof. Dr.Adelardo Adelino Dantas de Medeiros
Departamento de Computação e Automação
UFRN

____

Prof. Dr. Luiz Affonso Henderson Guedes de Oliveira
Departamento de Computação e Automação
UFRN

Dedico este relatório aos meus pais Gentil Aquino e Maria Neta, pelo apoio e educação proporcionado. E ao meu filho que tem dado motivação para a conclusão deste trabalho.

Agradecimentos

Agradeço grandemente :

– primeiramente a Deus;

– aos meus pais, Gentil Aquino e Maria Neta, minhas irmãs Verônica e Helenilda, por estarem comigo e apoiarem em todos os momentos.

– à minha esposa Adriana e meu filho César, pelo carinho e compreensão.

– às minhas Avós Rita e Helenilda, pelas orações e demonstrações de carinho.

– à Profa. Thaís, pela orientação, incentivo e paciência em mim depositados.

– a Alysson, pela ajuda nos códigos JAVA e SOAP.

– à Agência Nacional do Petróleo, pelo suporte financeiro para a execução deste trabalho.

– à Universidade Federal do Rio Grande do Norte, por dispor sua estrutura física para meu aprendizado.

– aos mestres, pela transmissão dos conhecimentos.

– aos amigos, pela companhia agradável nesses anos de convivência.

Resumo

Uma Interface Web é capaz de publicar grande quantidade de informações de forma centralizada e interativa para o usuário, plotar gráficos da Carta Dinamométrica e outras variáveis interessantes ao monitoramento de poços de petróleo. A interface web deverá estar disponível em uma intranet. Esse fato, associado a mecanismos de controle de acesso de usuários de diferentes níveis confere segurança ao sistema e, ao mesmo tempo, possibilita que as informações sobre os poços de petróleo possam ser acessadas a partir de qualquer ponto da intranet.

Permitir a análise desses dados sobre os poços, de forma interativa, torna possível a tomada de decisões importantes a respeito de viabilidade, custos de operação e manutenção de equipamentos.

Este trabalho tem como objetivo apresentar uma interface web para um sistema distribuído de monitoramento em tempo real de reservatórios de petróleo com elevação artificial. Foi utilizado o protocolo de Web Services, Soap que permite a comunicação de sistemas desenvolvidos em diferentes linguagens através da linguagem XML. Além disso o site proporciona uma interface visualmente mais interessante ao acesso das informações, e um controle de acesso seguro.

Sumário

Lista de Figuras

Lista de Tabelas

Acrônimos

CORBA	Common Object Request Broker Architecture
CLP	Controlador Lógico Programável
CSS	Cascaded Style Sheet
DOC	Distributed Object Computing
FTP	File Transfer Protocol
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IDL	Interface Definition Language
IIOP	Internet Inter-ORB Protocol
OMG	Object Management Group
ORB	Object Request Broker
RPC	Remote Procedure Call
SOAP	Simple Object Access Protocol
SMTP	Simple Mail Transfer Protocol
TCP/IP	Transfer Control Protocol/Internet Protocol
WWW	World Wide Web
W3C	World Wide Web Consortium
XML	eXtensible Markup Language
XSD	XML Schema Definition

1 Introdução

A exploração terrestre nacional de petróleo se dá em sua maioria por Bombeio Mecânico com Hastes [1], correspondendo a 60% da produção terrestre nacional. Este método de extração corresponde então a uma parcela considerável da produção nacional de petróleo que, acompanhada de um sistema de monitoramento, poderá obter melhor controle no processo de exploração. Fato que justifica a importância de que este processo de produção funcione com alta eficiência e seja apoiado por ferramentas computacionais amigáveis que facilitem o monitoramento, a visualização e a alteração das informações referentes ao processo.

Através de um sistema de monitoramento um usuário pode observar o processo de produção, avaliá-lo e estabelecer ações a serem realizadas sejam elas corretivas, preventivas ou que visem otimizar o processo. Portanto, o monitoramento pode ser visto como uma poderosa ferramenta que acompanha o desempenho da produção e visa assegurar a melhoria do nível e qualificação dessa atividade.

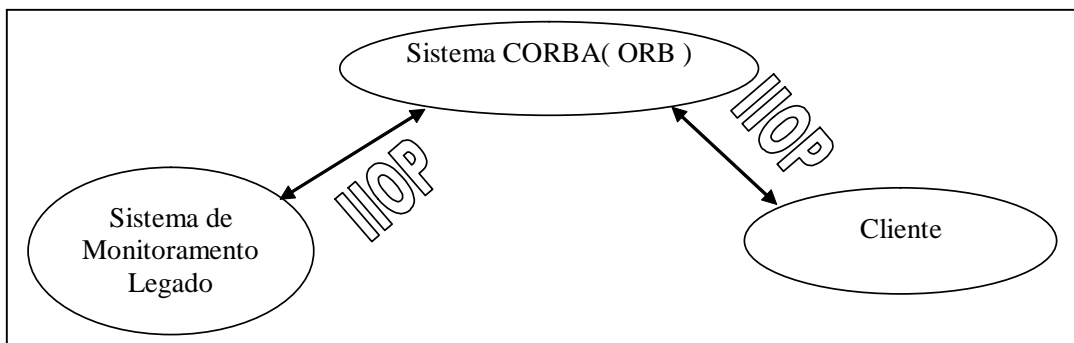
Um sistema de monitoramento que permita a visualização atualizada da exploração de petróleo em poços remotos é primordial à empresa exploradora uma vez que pode guiar decisões importantes visando uma maior produtividade. Este sistema deve permitir a visualização de variáveis da produção, do estado do poço e da vida útil do mesmo. Além disso, deve dar suporte a algumas ações como o ligamento/desligamento do bombeio e o estabelecimento de valores para o *idletime*. Com essas facilidades, a equipe responsável teria um suporte computacional para minimizar custos e potencializar a produção nesse tipo de processo de exploração de petróleo. O sistema deve também garantir que apenas usuários autorizados tenham acesso as informações e que diferentes classes de usuários tenham visões distintas do sistema. Por exemplo, um engenheiro de produção deve ter

acesso a determinadas informações que não podem ser vistas pelo técnico. Outro aspecto importante é a fácil acessibilidade ao sistema em diferentes locais de acesso da mesma empresa. Para isso, é interessante permitir acesso ao sistema através de uma Intranet via *browser*.

1.1 O Sistema Legado

O trabalho descrito em [2] definiu e implementou uma arquitetura computacional que oferece suporte ao monitoramento de poços distribuídos. O sistema concentrou-se no desenvolvimento da estratégia de comunicação entre as partes integrantes do sistema remoto. Essa arquitetura utiliza o padrão CORBA (*Common Object Request Broker Architecture*) [3], uma plataforma de *middleware* que facilita o desenvolvimento de aplicações distribuídas provendo um protocolo de comunicação que torna transparente a distribuição e a heterogeneidade de linguagens e plataformas de execução. O trabalho disponibiliza um sistema computacional que permite a instrumentação em cada poço exibindo as variáveis importantes e o vetor de pontos que compõem a carta dinamométrica. A interface com o usuário é textual o que limita fortemente a usabilidade do sistema em especial para os usuários com pouco conhecimento computacional. Ainda em [2] a comunicação entre o cliente e o sistema de monitoramento, se dá através do CORBA (ORB) que aguarda conexões de usuários e gera um número ID que o identifica no servidor para troca de mensagens durante a comunicação. Este processo se dá através de mensagens assíncronas utilizando o protocolo IIOP (*Internet Inter-ORB Protocol*) como é ilustrado na Figura 1.1.

Sistema CORBA(ORB)
IIOP
IIOP
Sistema de Monitoramento Legado
Cliente

Figura 1. 1 – Arquitetura do Sistema Legado.

Como mencionado anteriormente, a interface com o usuário é textual. A interação textual disponibilizada para o monitoramento, que poder ser vista na Figura 1.2, dificulta bastante o seu uso e restringe o acesso às

maquinas que possuem o sistema instalado. Além disso, o gráfico correspondente a carta dinamométrica não é exibido. São exibidos apenas os elementos do vetor que a representa.

```
Prompt de comando - consumer.exe -ORBInitRef NameService=iiop://localhost:2809/NameService

****MONITORAMENTO DE POCOS COM ELEVACAO ARTIFICIAL******
  --Campo--          --N Pocos--

0 - Guarulhos              27
1 - Barao                  43
2 - Itai                   31
3 - Guernica               25
4 - Mare                   46
5 - Poio                   43
6 - Arautu                 39
7 - Galinhos               11
8 - Arabaina               8
9 - Oriche                 2
10 - Black                  25
11 - Simpaio                14
12 - Natal                  9
13 - Tambore                11
14 - Chui                   24
15 - Iapoque                43
16 - Iracema                39
Entre com a opcao desejada

---- 1 : Visualizar o Idle_Time
---- 2 : Ultima Atualizacao dos dados
---- 3 : Visualizar Carta Dinamometrica
---- 4 : Visualizar Carta Dinamometrica Atual
---- 5 : Estado do Bombeio Mecanico
---- 6 : Setar o Idle_Time
---- 7 : Ligar/Desligar o Bombeio Mecanico
---- 8 : SAIR do Programa
 -->_
```

Figura 1. 2 - Inteface do Sistema de Monitoramento Legado.

Outro ponto falho do sistema legado está na ausência de mecanismos de segurança e confidencialidade das informações. Esse é um aspecto essencial nesse contexto de monitoramento da produção de petróleo uma vez que os dados da produção são extremamente confidenciais. Portanto, o acesso às informações do processo de exploração deve ser exclusivo às pessoas autorizadas. Dessa forma, é necessário que o sistema computacional disponha de um serviço de autenticação.

Uma outra limitação é o fato de não haver distinção entre diferentes classes de usuário. Dessa forma, todo o usuário do sistema tem acesso a todas as informações. No entanto, há necessidade de se distinguir as diferentes classes de usuários e permitir que os mesmos tenham diferentes visões das mesmas informações – usuários que ocupam posição de gerenciamento do processo produtivo devem ter uma visão abrangente dos dados de produção e devem poder determinar ações que não podem ser determinadas

por outros usuários.

1.2 Objetivo do Trabalho

O objetivo desse trabalho é desenvolver uma interface web para o sistema de monitoramento apresentado acima incluindo um mecanismo para autenticação de usuários e distinção entre diferentes classes de usuários, proporcionando diferentes visões dos dados do processo de exploração de petróleo. A visualização gráfica da carta dinamométrica também deve fazer parte dessa interface.

A concretização desse objetivo consiste na migração do sistema de interface textual para o ambiente web onde, através de um *browser*, seja possível interagir com o sistema de monitoramento. Para o acoplamento com o sistema de monitoramento existente surgiu a necessidade de usar o protocolo SOAP (*Simple Object Access Protocol*) [4] devido ao fato deste ser um padrão da W3C para troca de documentos e comunicação remota, efetivando assim a interoperabilidade entre os sistemas.

1.3 Estrutura do Trabalho

Este trabalho é composto por em cinco capítulos. O primeiro deles contém uma breve introdução de como este foi desenvolvido, qual o escopo de sua aplicação, problemas encontrados e uma breve descrição do que foi feito para a solução destes problemas.

No segundo capitulo é feita uma fundamentação teórica sobre o monitoramento de poços, sobre a arquitetura proposta pelo Sistema Legado e sobre as tecnologias utilizadas no desenvolvimento do sistema Web.

O terceiro capítulo traz detalhes da estratégia de desenvolvimento do trabalho, apresenta a arquitetura do sistema distribuído acoplado a interface web e sua implementação. São esclarecidos os aspectos de funcionalidade como o acoplamento entre as partes do sistema legado, do Web Service, da autenticação do usuário, do

acesso ao banco de dados e detalhes do projeto do sistema web.

O quarto capítulo trata de trabalhos relacionados ao mesmo escopo de utilização deste. São descritos dois sistemas de monitoramento de petróleo e comparados com o presente trabalho.

Por último o quinto capítulo apresenta a conclusão do trabalho. São evidenciados as contribuições e os possíveis avanços neste trabalho que aprimorariam ainda mais o sistema de monitoramento.

2 Conceitos Básicos

2.1 O Sistema Distribuído Legado de Monitoramento de Poços

2.1.1 Características do Processo Supervisionado

O processo monitorado abrange o conjunto de poços perfurados em reservatórios de petróleo que utilizam o bombeio mecânico com hastes como método de elevação artificial. É importante saber o que cada variável de monitoramento representa neste sistema para entender melhor o processo por completo. As variáveis de maior importância para serem monitoradas durante o bombeio foram a *carta dinamométrica*, e o *idle-time*.

Nos reservatório de petróleo explorados com Unidades de Bombeio Mecânico (UBM), a vazão do reservatório é geralmente inferior a vazão do equipamento instalado para o bombeio do fluido. Ocorre que para evitar o funcionamento da UBM no poço temporariamente vazio, é estabelecido um tempo para este equipamento permaneça parado. Esse tempo deve ser suficiente para a vazão do reservatório restabeleça o local de exploração do poço. O tempo em que o equipamento de bombeio permanece parado é denominado ***idle time*** e o tempo em que este permanece em efetivo bombeio se chama ***run time*.** Estes dois períodos de tempo são determinantes na eficiência da produção e somados formam um ciclo de produção ou ***cycle time*** [2].

Na UBM, a tensão na coluna de sua haste e a posição relativa ao bombeio estão sendo medidos várias vezes por segundo, estes valores são levados a um computador que, além de traçar um gráfico, que leva o nome de ***carta dinamométrica***, calcula a quantidade de trabalho realizado. A carta dinamométrica é a principal ferramenta disponível para avaliação das condições em que está ocorrendo o bombeio. A carta é

obtida instalando-se um dinamômetro para registrar as cargas na haste polida durante um ciclo completo [1]. Com a informação do trabalho realizado pode-se constatar por exemplo problemas como o *pump off*, que corresponde ao funcionamento da bomba com seu enchimento parcial, o que pode danificar vários elementos da UBM. Um exemplo de carta dinamométrica pode ser visualizado na Figura 2.1.

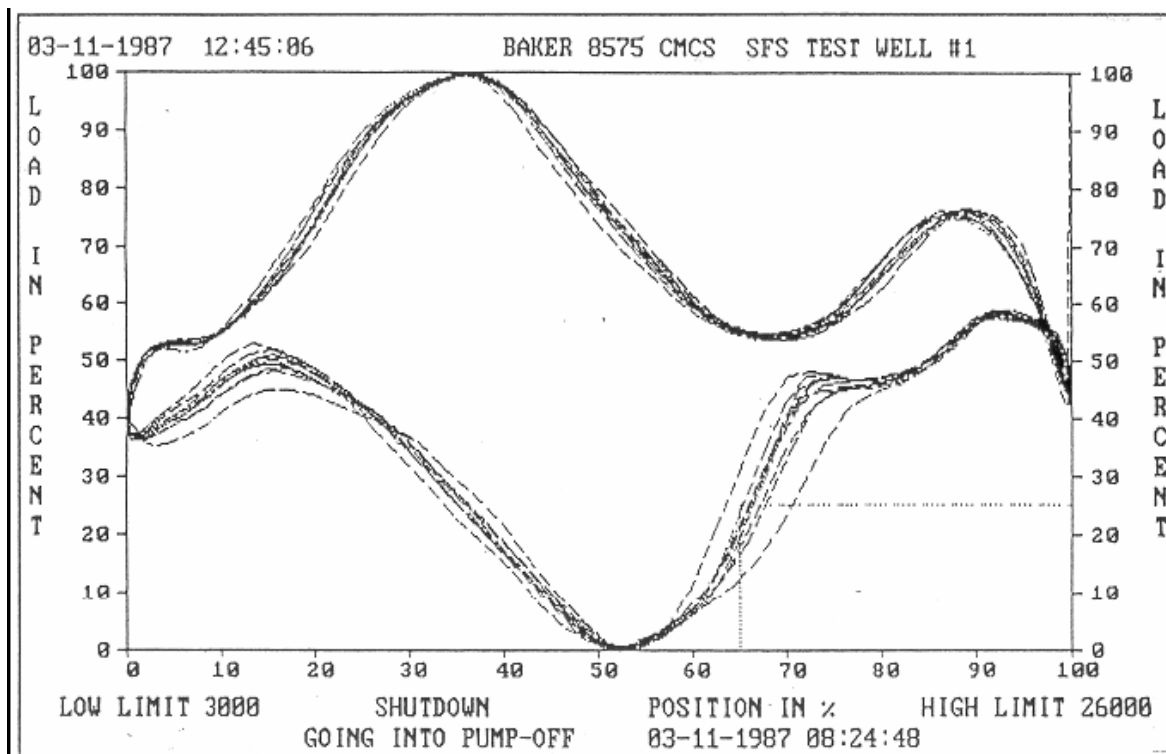



Figura 2. 1 – Carta Dinamométrica

2.1.2 O Sistema de Monitoramento

O Sistema foi idealizado para supervisionar poços de petróleo automatizados por bombeio mecânico nos campos da Petrobrás da Bacia Petrolífera Potiguar que abrange Rio Grande do Norte e uma parte do Ceará, uma vez que o estado que dá nome a bacia é o possuidor da 2ª maior exploração de petróleo Nacional, ficando atrás somente da Bacia de Campos. Se analisado somente a exploração terrestre é o maior produtor nacional [5]. Os sensores de posição, a célula de carga e um *Controlador Lógico Programável* (CLP), correspondem a todo o hardware instalado localmente no poço e são responsáveis respectivamente pelas medições de posição, de carga, e pelo controle do processo. O

CLP além de receber os sinais dos sensores e outras informações relativas ao bombeio, como hora/data de coleta dos dados, disponibiliza-os para leitura através da porta de comunicação. O meio físico utilizado é o enlace de rádio, que através dos radio modens transmite as informações coletadas.

Com a finalidade de realizar a comunicação entre os CLP's e uma central de supervisão foi estabelecida uma Rede de Comunicação de Campo. Um computador PC efetua os comandos de supervisão e armazena as informações dos poços monitorados. A arquitetura da rede é do tipo mestre-escravo e as informações são transmitidas por enlace de rádio.

Um importante elo entre as partes do sistema é a **central de monitoramento (CM)** que permite a comunicação, via enlace de rádio, com os CLP´s dos poços e também com outras máquinas. Essa central realiza um procedimento de varredura de campo em todos os poços, colhendo suas variáveis e armazenando-as em um banco de dados. Os usuários de uma rede local interligada a CM tem acesso as informações armazenadas através de uma comunicação entre as máquinas da rede local e a CM, comunicação esta realizada via CORBA, de forma transparente.

2.1.2.1 Variáveis do Processo de Bombeio Mecânico

As tarefas controladas pela central de Monitoramento são: Varredura automática, Monitoração do Sistema e Ajuste de Variáveis do Bombeio, tendo como objetivo deixar que esse controle seja feito por outras máquinas pertencentes a rede, remotamente, via CORBA.

A **varredura automática** é uma tarefa realizada periodicamente buscando todas as informações de cada poço que são ligados a um CLP.

A **monitoração do sistema** é uma tarefa realizada conjuntamente com a anterior realizando o processamento das informações que acabaram de chegar e tomando alguma decisão como acionar um alarme ou modificar parâmetros do controlador.

O **ajuste de variáveis do bombeio** realiza o ajuste das duas principais variáveis sujeitas a alterações: o *idle_time* e o *run_time*. Este ajuste busca elevar a extração do óleo e reduzir o consumo de energia do cavalo mecânico, otimizando o funcionamento dos poços.

2.1.2.2 A Arquitetura do Sistema

A arquitetura de Sistema de Monitoramento está ilustrada na Figura 2.2 O acesso via web, ilustrado nessa figura, é a nova parte definida no presente trabalho.

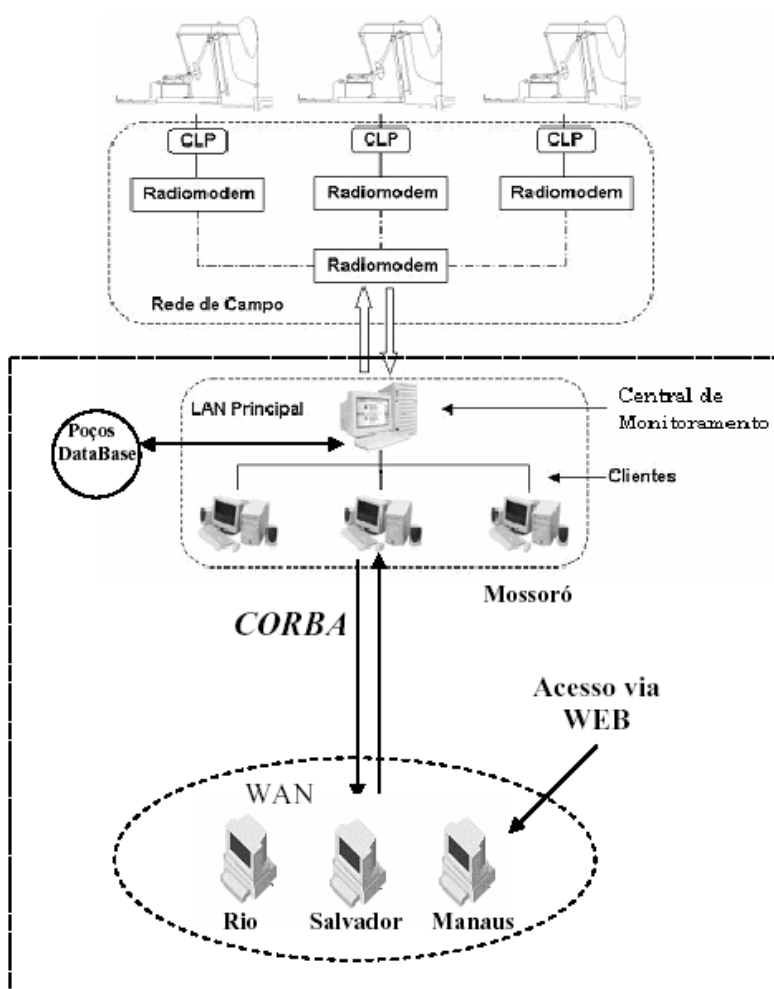



Figura 2. 2 – Arquitetura do Sistema de Monitoramento.

Com relação ao modelo de implementação do sistema foi definida a arquitetura cliente-servidor. Esta escolha é justificada pelas análises feitas em comparação a outras disponíveis, que comprovou ser a que melhor atende aos requisitos deste sistema de

monitoramento. Definida a arquitetura do sistema distribuído a ser implementada, foram desenvolvidas duas estratégias distintas.

A primeira dessas, denominada *estratégia 1*, traz como principal característica prover um sistema de monitoramento que além de visualizar parâmetros de monitoramento, pode interagir com o sistema atuando em variáveis de controle ou executando comandos ou ações úteis de serem executadas após analisados dados adquiridos do processo de exploração dos poços em questão.

A segunda estratégia de desenvolvimento, denominada *estratégia 2*, é mais objetiva e permite ao usuário somente visualizar variáveis importantes no processo de extração de petróleo privando-o de intervir no sistema de monitoramento. Esta foi desenvolvida após a primeira, porém não foi esta que foi tomada como base para nesse trabalho em relação ao desenvolvimento do sistema Web.

A estratégia 1 foi a escolhida para ser tomada como ponto de partida para o avanço da Interface Web pois é a opção mais completa disponibilizando a visualização das variáveis e permitindo a intervenção no sistema quando necessário. Esta estratégia implementa um cliente CORBA que tem duas funções principais: a primeira é garantir a conexão ao usuário conectado ao sistema, pois é através deste que as requisições serão elaboradas e impostas ao cliente; a segunda função realiza e mantém a comunicação com o servidor CORBA. .

Algumas tarefas realizadas pelo cliente são transparentes ao usuário para garantir a comunicação com o servidor, são elas:

- Busca do objeto servidor via serviço de nomes
- Etapa de identificação do usuário
- Requisição de Tarefas
- Término da Comunicação

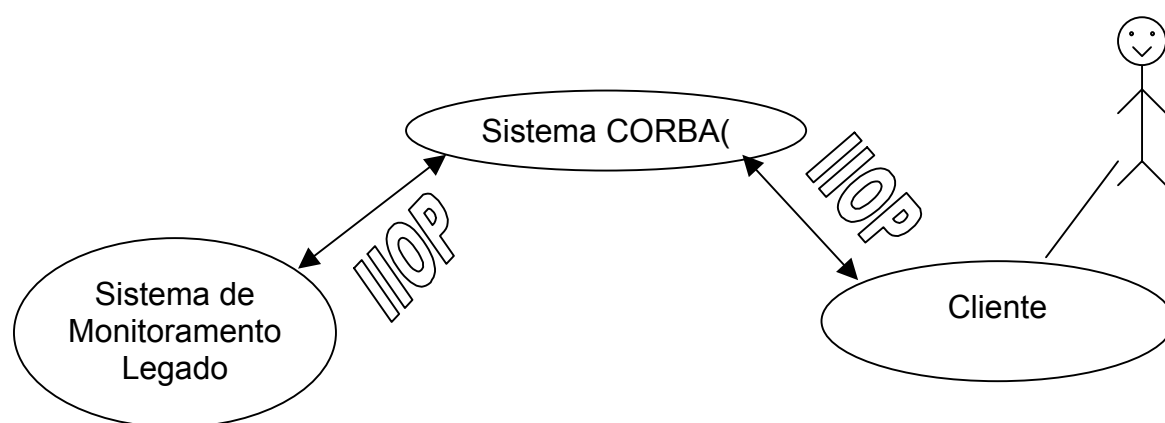



Figura 2. 3 –Comunicação do Usuário - Sistema Legado.

O servidor CORBA, como descrito na Figura 2.3, corresponde a *central de monitoramento*, responsável pela captura das informações vindas dos poços monitorados além de efetuar os comandos de supervisão desses poços. Este servidor deve ainda funcionar como elo de ligação com o cliente e desempenhar algumas tarefas necessárias para garantir um sistema de comunicação estável com o cliente. Algumas dessas tarefas referem-se a constante supervisão e armazenamento dos dados dos poços monitorados e, ainda, a recepção das requisições dos clientes, execução de suas tarefas e fornecimento das informações de conclusão da tarefa requisitada.

Os parâmetros usados pelo sistema, expressos na sua interface IDL, estão representados na Tabela 2.1.

VARIÁVEL	DESCRIÇÃO
Run_time	Tempo que o bombeio mecânico de um dado poço fica operando
Idle_time	Tempo que o bombeio mecânico de um dado poço fica parado
Ultima_atualizacao	Momento da última atualização dos dados de um específico poço
Carta_dinam	Carta dinamométrica, conjunto de 128 pares de pontos lidos
Status_bombeio	Define se o bombeio está acionado ou desligado
Campo	Número de identificação do campo de petróleo monitorado
Poço	Número de identificação do poço de petróleo monitorado
Nome_campos	Array de nomes dos campos de uma central de monitoramento
Task	Identificador das tarefas a serem realizadas
Tag_	Auxilia no processo de troca de mensagens
ID_Remetente	Número único de identificação do cliente cadastrado

Tabela 2. 1– Parâmetros definidos na Interface IDL da estratégia adotada.

Por fim, vale salientar que neste cliente CORBA implementado, a coleta das requisições do usuário se dá através de uma interface no modo texto, pouco amigável. Nesta, primeiramente são listados todos os campos que estão implementados, seu respectivo nome e número de poços como pode ser observado na Figura 2.4. Em seguida, através de um menu principal, o usuário escolhe, dentre algumas opções, a que gostaria de submeter ao servidor. Na seqüência, o usuário escolhe o campo e o poço a ser analisado. A seguir a requisição é enviada ao servidor que se encarrega de dar a resposta.

```
Prompt de comando - consumer.exe -ORBInitRef NameService=iiop://localhost:2809/NameService

****MONITORAMENTO DE POCOS COM ELEVACAO ARTIFICIAL******
   --Campo--           --N Pocos--

0 - Guarulhos             27
1 - Barao                 43
2 - Itai                  31
3 - Guernica              25
4 - Mare                  46
5 - Poio                  43
6 - Arautu                39
7 - Galinhos              11
8 - Arabaina              8
9 - Oriche                2
10 - Black                 25
11 - Simpaio               14
12 - Natal                 9
13 - Tambore               11
14 - Chui                  24
15 - Iapoque               43
16 - Iracema               39
Entre com a opcao desejada

---- 1 : Visualizar o Idle_Time
---- 2 : Ultima Atualizacao dos dados
---- 3 : Visualizar Carta Dinamometrica
---- 4 : Visualizar Carta Dinamometrica Atual
---- 5 : Estado do Bombeio Mecanico
---- 6 : Setar o Idle_Time
---- 7 : Ligar/Desligar o Bombeio Mecanico
---- 8 : SAIR do Programa
 -->_
```

Figura 2. 4 – Interface Modo Texto do Usuário.

2.2 O Protocolo SOAP

Programas distribuídos, em geral, atuam em ambientes com uma variedade de plataformas e ambientes com características heterogêneas. Devido à heterogeneidade é necessário a adoção de um padrão para garantir a interoperabilidade entre os elementos distribuídos.

Algumas plataformas como CORBA, DCOM e RMI, foram propostas com essa finalidade. No entanto, elas apresentam algumas incompatibilidades entre si que implicam em restrições e aumentam a complexidade de desenvolvimento de sistemas

nessa área. Alem disso, originalmente nao incluem facilidades para acesso aos objetos via browser web. Para que esse acesso seja implementado é necessário um suporte de comunicacao entre tais plataformas e o protocolo HTTP (*Hypertext Transfer Protocol*) [8]

2.2.1 A Evolução do SOAP

O SOAP evoluiu de uma tentativa inicial de definir um modo de enviar chamadas de método e parâmetros de um computador para outro, chamada XML-RPC (*Remote Procedure Call*). Essa especificação inicial foi definida em 1998 por Dave Winer, de uma empresa chamada UserLand. A IONA, a Microsoft e a IBM se interessaram em melhorar o método XML-RPC. Esse novo método se tornou a especificação SOAP 1.1 [6]. Inicialmente com o significado de *Simple Object Access Protocol,* no entando esta designação foi perdendo o significado e SOAP passou a ser entendido como um protocolo de troca de mensagens. Em 2000 este protocolo foi submetido ao *World Web Consortium (W3C)*, onde atualmente está disponível a recomendação SOAP 1.2. Na terminologia usada pelo W3C uma recomendação corresponde ao mais alto status de uma especificação. É importante ressaltar ainda que é política da W3C evitar a palavra “padrão” uma vez que eles são um consórcio e não um comitê de padrões.

2.2.2 O Que é SOAP

É um protocolo construído sobre XML, o que significa que os documentos SOAP são documentos XML, para dar suporte a comunicação remota entre sistemas distribuídos ão são especificados os tipos de dados que podem ser transferidos ou quais chamadas de função podem ocorrer sobre ele. Dessa forma pode-se dizer que um cliente SOAP simplesmente formata um arquivo de texto e o transfere para a outra máquina.

A grande vantagem do SOAP é oferecer interoperabilidade entre um ampla variedade de plataformas, isto permite prover uma especificação generalizada para

chamada de métodos em objetos e componentes usando chamadas via protocolo HTTP e documentos no formato XML(*eXtended Markup Language*). Um ponto a ser observado é que apesar de preferido, o HTTP não é uma forma obrigatória de envio de mensagens SOAP, o transporte dessas mensagens também pode se dar por protocolos outros como SMTP ou FTP. Outro aspecto importante é que o SOAP não é restrito a um modelo de objeto especifico, o que permite que clientes desenvolvidos em um determinado ambiente possa acessar métodos implementados em outro ambiente qualquer.

As primeiras versões de SOAP disponibilizavam como unica opção para o meio de transporte o protocolo HTTP. Uma das desvantagens do HTTP é que não há garantia de entrega dos dados enviado., As novas versões ampliaram as possibilidades para outros protocolos como o FTP e o SMTP.

2.2.3 A Gramática SOAP

Na gramática SOAP o acesso a objetos se dá pela chamada de métodos. Uma mensagem SOAP traz uma estrutura bem simples de ser implementada. As partes obrigatórias são o envelope e o corpo SOAP, existe ainda uma parte opicional - o cabeçalho SOAP. Um fator interessante de identificação de mensagens que usam este protocolo é que as *tags* do documento XML associadas ao SOAP possuem o prefixo soap-env, no SOAP 1.2, abordado neste trabalho.

Uma mensagem SOAP traz um envelope que é definido pela *tag SOAP-env:Envelope*. Neste está contido o cabeçalho *SOAP-env:Header*, e existe ainda o corpo cuja *tag* é *SOAP-env:Body*. A Figura 2.5 ilustra a relação entre as partes de uma mensagem SOAP.

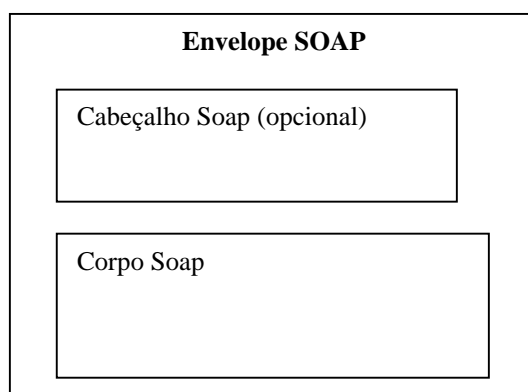



Figura 2. 5 – O envelope SOAP contendo cabeçalho e corpo

O inicio de uma mensagem SOAP é conhecido ao se encontrar a tag:

<soap-env:Envelope>

A seguinte tag termina o envelope:

</soap-env:Envelope>

No caso de SOAP, suas mensagens não podem ser enviadas em lotes, logo apenas uma mensagem existe dentro de um envelope. Pode haver ainda uma seção de cabeçalho que pode conter n elementos de cabeçalho. Na prática, a tag *SOAP-env:Envelope* é implementada da seguinte forma

<soap-env xmlns:SOAP-env="http://www.w3.org/2003/05/soap-envelope">

O termo *xmlns* é uma palavra-chave da XML que é usada para identificação única das *tags* com a finalidade de evitar conflitos de nomes. *SOAP-env* é o nome exigido como prefixo para todos os nomes de tags definidos pelo SOAP. A string "http://www.w3.org/2003/05/soap-envelope" é uma string unica cuja finalidade é indicar a versão do soap que este cliente está usando. Com isso, um web service que receba a requisição pode analisar esta string e determinar se ele é capaz de se comunicar usando essa versão do SOAP.

Dentro de uma mensagem SOAP a tag *SOAP-Env:Body* indica o início do corpo da mensagem: <soap-env:Body>

A finalização do corpo é identificada pela tag

</soap-env:Body>

Neste local coloca-se a parte a ser executada da mensagem, normalmente uma chamada remota de método, com valores dos parâmetros. Pode ser também que esteja simplesmente a transferência de um documento XML .Pode ainda acontecer de nessa estrutura de corpo passar outro tipo de dados como uma imagem, um vetor ou ainda um valor monetário, o formato deste corpo é controlado pelo desenvolvedor do Web Service.

O exemplo a seguir ilustra um segmento que executa uma chamada remota a um método chamado *getidletime()*:

```
<soap-env:Body>
<soap:Body>
<x:MyData xmlns:x='http://example.org/getidletime()'>
<x:campo>1</x:name>
<x:poco>2</x:name>
<x:img src='http://example.org/me.png'/>
</x:MyData>
</soap:Body>
</soap-env:Body>
```

Tag SOAP-Header

Este elemento é opicional em uma mensagem SOAP mas, quando presente, deve ser o primeiro elemento dentro de um envelope SOAP.

Seus principais atributos associados são: *SOAP-env:mustUndestand* e *SOAP-env:actor*. O primeiro refere-se a necessidade de restringir o acesso do serviço à mensagens capazes de manipular campos deste cabeçalho. Então as mensagens com tal atributo setados em “1” só podem ser manuseadas por Web Service capazes de processar seu cabeçalho. Caso contrário, uma mensagem de erro será gerada.

O segundo atributo, *SOAP-env:actor* define a URI (equivalente a URL usada no HTTP) a qual o cabeçalho que se refere. Ele é usado para encadeamento de Web

Services necessários para que este serviço seja completamente processado - tarefa comum em transações comerciais.

Um exemplo de cabeçalho pode ser visto abaixo.

```
<SOAP-env:Header>
<!--definição do namespace como sendo "meuNS" para personalizar o documento SOAP-->
<meuNS:mercado xmlns:meuNS="http://www.mercadim.br/valores/" soap:actor=
"http://www.mercadim.br/descricao/" SOAP-ENV:mustUnderstand="1"/>
<meuNS:lingua>port</meuNS:lingua>
<meuNS:dinheiro>REAL</meuNS:dinheiro>
</meuNS:mercado>
</SOAP-ENV:Header>
```

Este cabeçalho possui um elemento fictício chamado *<meuNS:mercado>*. Esse elemento contém os elementos língua e dinheiro, elementos estes que não possuem qualquer significado padrão no SOAP atualmente. O fato de SOAP não suportar alguns recursos muito necessários como sessões, transações e autenticação de usuários tem sido alvo de críticas, mas isto não impede o desenvolvedor de usar seus próprios métodos para sanar problemas de autenticação.

2.2.4 Relatando Erros ao Cliente

Em um sistema de tratamento de erros deve ser informado à equipe de suporte o máximo de informação possível sobre o erro ocorrido. O método utilizado por SOAP para tratamento de seus erros é baseado no uso da tag *soap-env:fault*. Esta tag é considerada filha da tag *soap-env:Body* na hierarquia da especificação e tem por finalidade informar que um problema ocorreu na tentativa de processamento da requisição enviada ao Web service. Esse elemento opcional apenas precisa aparecer nas mensagens de resposta e ainda uma unica vez por mensagem. As seguintes tags são opcionais dentro da tag *soap-env:fault*:

soap-env:faultcode - Este elemento é exigido pela especificação e deve conter algum código indicando qual é o problema causador deste erro.

soap-env:faultstring - Este elemento é uma versão legível ao usuário; é uma breve explanação do erro, mostrando o código do erro ocorrido.

soap-env:faultactor - Elemento opicional informa que serviço gerou a falha. Importante quando o sistema utiliza uma cadeia de serviços para acessar a requisição.

soap -env:detail - Esse elemento contém o máximo de informações disponíveis sobre o estado do servidor no momento que ocorreu a falha.

Em SOAP os códigos de erros são representados de maneira peculiar, strings de duas partes, com códigos de erro principais e secundários separados por um ponto:

```
<soap-env:faultcode>
server.customerCreatedFailed
</soap-env:faultcode>
```

Geralmente são definidos apenas quatro tipos genéricos de especificação de erro, são elas:

- *Server* – este ocorre quando um erro no servidor é responsavel pelo problema, mas a mensagem está correta. Quando o cliente está sendo escrito, geralmente é implementado para repetir a tentativa de mensagens que falham com este código de erro, visto que um erro erro com a disponibilidade do serviço pode, na maioria das vezes, ser sanado com uma nova tentativa. No entanto é conveniente limitar o numero de tentativas pois um erro do próprio serviço, não seria solucionado com o passar do tempo.

- *Client* – indica que o erro está não mais no servidor e sim na mensagem enviada, pode ser um formato errôneo, uma mensagem incompleta, etc.

- *versionMisMatch* – este erro ocorre quando versões dos protocolos do cliente e do servidor são diferentes. A versão é determinada pelo URI de namespace usado na tag *soap-env:Envelope*.

- *mustUnderstand* – erro gerado quando um elemento no cabeçalho estiver marcado como obrigatório e não poder ser processado. Sempre que no cabeçalho a tag *mustUnderstand* está marcada com "1", exige-se do web service a capacidade de interpretar o conteúdo deste elemento.

2.2.5 Tipos de Dados do SOAP

A especificação do SOAP define o suporte aos tipos de dados baseado no XSD (XML Schema Definition language). Esta especificação define padrões para a descrição de tipos primitivos de dados assim como estruturas complexas e hierárquicas. Tipos definidos pelo usuário também são aceitos, de forma que é possível formar estruturas de dados a partir dos dados primitivos oferecidos pela XSD. Uma importante vantagem do uso do SOAP é aceitar qualquer tipo que possa ser representado por um XSD Schema. A Tabela 2.2 ilustra os tipos básicos de dados suportados pelo SOAP.

Boolean	Float	timeInstant
Byte	int	unsignedByte
Double	Long	unsignedInt
Datatype	Qname	unsignedLong
Decimal	Short	unsignedShort
Enumeration	String	

Tabela 2. 2 – Tipos básicos de dados suportados pelo SOAP.

3 Sistema Web para Monitoramento de Poços

3.1 Arquitetura

Na nova arquitetura proposta o **Web Service** é justamente o serviço prestado pelo servidor SOAP, este deve se comunicar com o cliente CORBA para enviar os dados que estão sendo solicitados, bem como recolher suas respostas. Para isso dever-se-ia alterar o cliente CORBA para que além de suas funções normais junto ao ORB, este pudesse aceitar a comunicação com a Interface Web do sistema. A partir daí este Web Service deve estabelecer conexões com **clientes** utilizando o mesmo protocolo, o SOAP, através de **sites de monitoramento**. Estes sites estão implementados em PHP, uma linguagem de script que oferece suporte a dinamicidade de páginas web. O sistema utilizando um mecanismo de autenticação de usuários cadastrados em um banco de dados PostGreSQL.

3.1.1 Problemas com o Sistema Legado

O processo de desenvolvimento do novo cliente apresentou dificuldades na tentativa de implementação deste com sua nova atribuição de permitir conexão ao Servidor SOAP. Primeiramente, no sistema legado, a versão compilada funcionava com uma pequena ressalva: quando o cliente CORBA fechava a conexão estabelecida com o servidor CORBA, ambos eram fechados. Apesar de não ser esse o esperado de um padrão de comunicação cliente-servidor, quando o cliente executava o procedimento de desconexão percebia-se a queda também do servidor. O código legado foi estudado,

revisto e ensaiado algumas alterações, sem êxito, pois sempre se deparava com algum problema de compilação.

3.1.2 Nova Versão da Aplicação CORBA

Para desviar deste problema, cresceu a expectativa de recriar a mesma aplicação. Procurou-se desenvolver de forma mais fiel possível a mesma arquitetura, as comunicações das partes, os algoritmos de simulação das variáveis adquiridas dos poços e inclusive no tocante a simulação de geração das cartas dinamométricas.

Analisando o código legado, foram criadas as mesmas estruturas de dados e seguiu-se a mesma estrutura de funcionamento do software. Muito embora, por conveniência do programador foi adotada a linguagem de programação JAVA, por uma maior familiaridade deste com a mesma, em relação à anteriormente utilizada, C++.

Foi desenvolvido um servidor CORBA, implementado na classe *MonitoraImpl*, e em seguida um cliente CORBA em modo texto, ambos desenvolvidos em linguagem JAVA utilizando a versão do ORB provida pela Sun.

Semelhantemente a versão anterior do sistema de monitoramento foi criada uma interface de modo texto para que os primeiros testes fossem feitos com relação a chamadas de funções pelo sistema distribuído, agora desenvolvido em linguagem JAVA. Este foi o novo ponto de partida para a implementação da Interface WEB de monitoramento de poços.

3.2 Implementação

Seguindo a arquitetura especificada anteriormente, esta seção descreve o procedimento para a implementação do Web Service e sua integração com a interface de monitoramento de poços. Inicialmente foi implementado o **Web Service**, para isso foi criado um **servidor** para as mensagens **SOAP**.

Este foi implementado em Java, e a classe onde a implementação citada acontece

é a *WsPetro.* Colocando em passos a comunicação deste Web Service temos:

Passo 1) Garantir uma comunicação deste **servidor SOAP** com o **cliente CORBA**;

Passo 2) Traduzir as chamadas recebidas por um **cliente SOAP,** para uma requisição ao cliente CORBA que a remeterá ao servidor CORBA.

Passo 3) Em seguida, o **servidor SOAP** deve aceitar a resposta do **cliente CORBA**, envia-la ao **cliente SOAP** para que este a publique no site que o acompanha.

Estes passos deixam explicitas as ligações entre os objetos da Figura 3.1 e ainda os protocolos utilizados em cada comunicação.

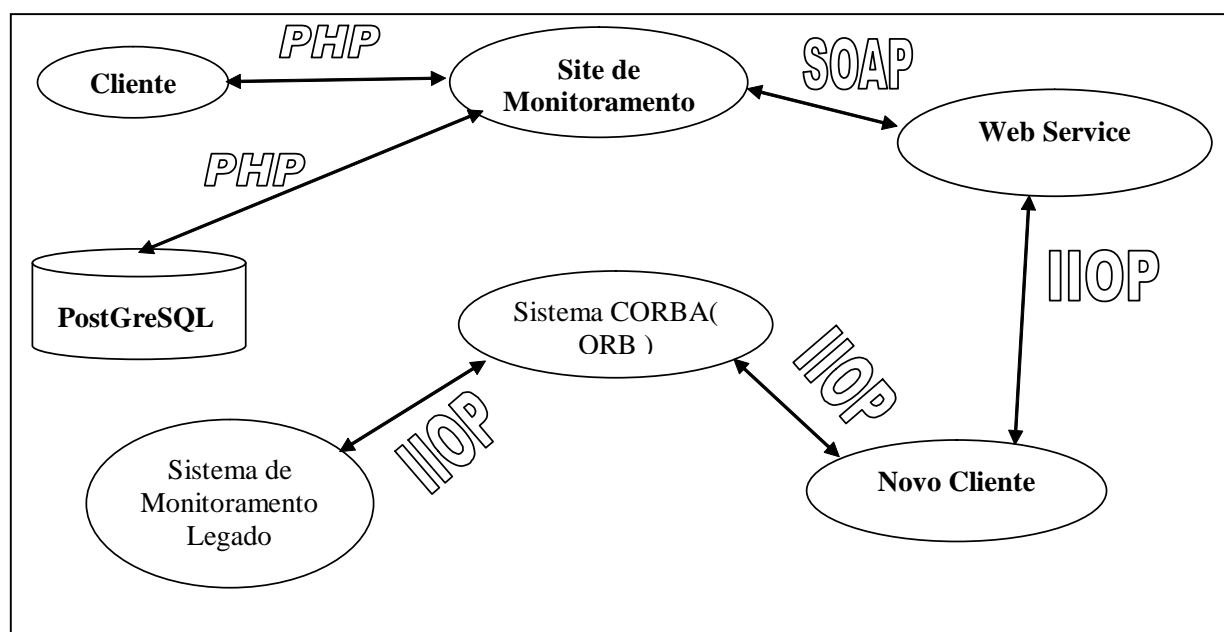



Figura 3.1 Arquitetura do Sistema Distribuído acoplado a Interface Web.

Todo este fluxo de requisições e respostas somente acontece quando o usuário entrar no site, sua autenticação seja conferida no cadastro armazenado em banco de dados. Caso esteja ele realmente cadastrado, é levado a uma página onde será montado um **menu** de opções específicas para seu perfil de acesso, é importante salientar que este perfil de acesso está intimamente ligado a sua posição hierárquica na companhia de exploração de petróleo. Estando habilitado poderá requerer informações que percorrerão todos os passos acima citados até que sejam publicados em um web site.

É possível observar nesta Figura 3.1 além da interligação entre as partes, os protocolos utilizados para a permuta de dados entre os distintos elementos.

3.2.1 O Servidor SOAP

O servidor SOAP serve de intermediário entre a comunicação do cliente SOAP e o cliente CORBA, para isso sua função é basicamente aguardar chamadas de seu cliente e traduzir esta chamada do envelope SOAP para o protocolo IIOP padrão ORB. O código deste servidor pode ser observado abaixo.

Pode ser observado na Figura 3.2 que a comunicação do servidor SOAP com o cliente CORBA se dá após o recebimento da requisição do cliente SOAP. Este exemplo mostra uma requisição do *idletime* do poço 3 do campo 2 do sistema de monitoramento. Em seguida o servidor em questão faz uma chamada ao serviço de nomes (nameservice), obtém como resposta a confirmação de conexão com uma referência do objeto Corba, envia a requisição do *idletime* do poço e recebe a reposta, passando-a logo em seguida ao cliente SOAP.

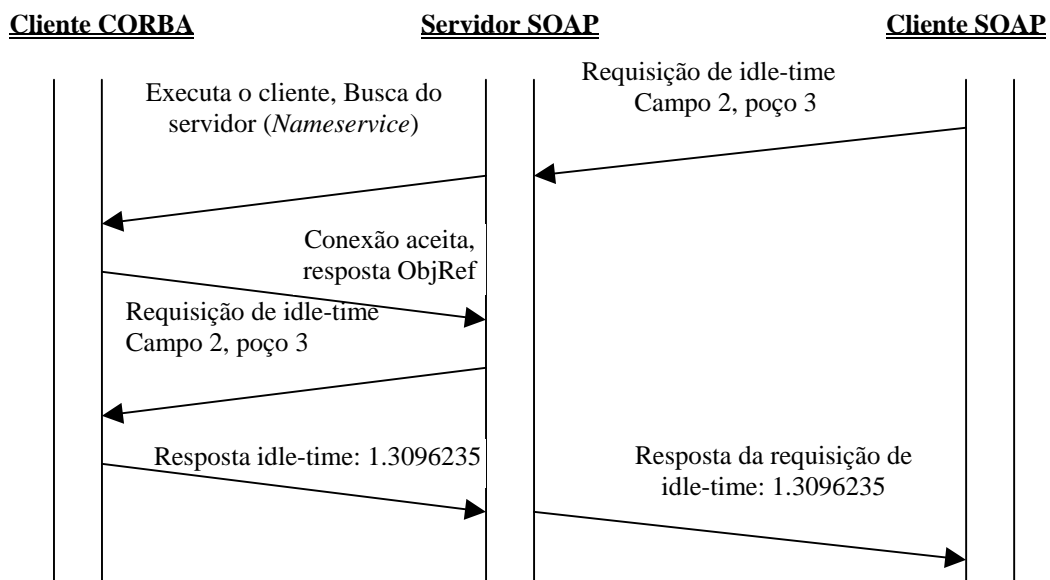



Figura 3.2–Estrutura Relacional e passos da comunicação.

O código da classe *WsPetro*, Figura 3.3, ilustra como este processo acontece. O programa retorna uma chamada à função *monitoraImpl.getIdleTime(indCampo, indPoco)*, passando o índice do campo e o índice do poço como parametros para esta função.

```
public class WsPetro{
	private Monitora monitoraImpl;
	public float getIdleTime(int indCampo, int indPoco){
		try{
			String args[]={"-ORBInitialPort", "1050" , "-ORBInitialHost", "localhost"};
			ORB orb = ORB.init(args, null);
			// get the root naming context
			org.omg.CORBA.Object objRef =
			orb.resolve_initial_references("NameService");
			// Use NamingContextExt instead of NamingContext. This is
			// part of the Interoperable naming Service.
			NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);
			// resolve the Object Reference in Naming
			String name = "Monitora";
			monitoraImpl = MonitoraHelper.narrow(ncRef.resolve_str(name));
		}catch (Exception e) {}
		return monitoraImpl.getIdleTime(indCampo, indPoco);
	}
```

Figura 3.3: Código do WsPetro.

3.2.2 O Cliente SOAP

O cliente do WebService, implementado na classe WsClient, cria para cada método da mensagem SOAP a ser enviada e as variáveis do envelope passado para o servidor. O código da Figura 3.4 ilustra a implementação da requisição do idle-time

```
public class WsClient{
	public static void main (String[] args) throws Exception {
		System.out.println("\n\nCalling the SOAP Server to getIdleTime \n\n");
		URL url = new URL (args[0]);
		//String name = args[1];
		Integer poco = new Integer(1);
		Integer campo = new Integer(1);
		Float idletime = new Float(3.5);
		Call call = new Call ( );
		call.setTargetObjectURI("urn:Example1");
		//Para o metodo IdleTime
		call.setMethodName("getIdleTime");
		call.setEncodingStyleURI(Constants.NS_URI_SOAP_ENC);
		Vector params = new Vector ( );
		params.addElement (new Parameter("Campo", Integer.class, campo, null));
		params.addElement (new Parameter("Poco", Integer.class, poco, null));
		call.setParams (params);
		System.out.print("The SOAP Server says IdleTime: ");
		Response resp = call.invoke(url, "");
		if (resp.generatedFault ( )) {
			Fault fault = resp.getFault ( );
			System.out.println ("\nOuch, the call failed: ");
			System.out.println (" Fault Code = " + fault.getFaultCode ( ));
```

```
                System.out.println (" Fault String = " + fault.getFaultString ( ));
        } else {
                Parameter result = resp.getReturnValue ( );
                System.out.print(result.getValue ( ));
                System.out.println( );
        }
...
```

Figura 3.4: Código do WsClient.

Com o cliente implementado, foi desenvolvida a página web que invoca este cliente e acessa às funções específicas de cada usuário. Para isso foi usada a linguagem *PHP*[9] para dar dinamicidade às funcionalidades do site, um banco de dados *PostGreSQL*[10] para o cadastro de usuarios no banco de dados e uma ferramenta de design de Web sites, chamada *Dreamweaver*[11]. Para fazer a chamada do cliente do Web Service foi encontrada uma ferramenta na Internet chamada NuSOAP[12], trata-se de um grupo de classes PHP de código aberto criada para consumir mensagens SOAP, esta não requer qualquer extensão do PHP e suporta o SOAP versão 1.1, justamente a versão utilizada no desenvolvimento desse sistema. A implementação de uma página usando o NuSOAP pode ser visualizada na Figura 3.5

```
<?php
        session_start("usuarios");
        include "conecta.php";
        …
        if($_SESSION[logado]=="sim"){
        $param = array();
        $serverpath ="http://10.9.96.158:8080/soap/servlet/rpcrouter";
        $namespace="urn:Example1";
        $client = new soapclient($serverpath);
        $resposta = $client->call("listaGeral",$param,$namespace);
        if (isset($fault)) {
                print "\nErro: ". $fault."\n"; }
        switch($_SESSION[perfil]){
        case "tecnico": ?>
        …
        case "estagiario": ?>
        …
        case "engenheiro": ?>
                <form method="post" action="resultit.php" name="idletime">
                <h1>Visualização de Idletime</h1>
                Sr(a) <?php echo $_SESSION[usuario] ?> informe o número do campo e o número do poço
                em questão: <br>
                <label for="campo">Campo : </label>
                <select name="selcampo">
                <?php for($i=0 ; $i<sizeof($resposta);$i++){?>
                <option value= <?php echo ($i+1) ?> > <?php echo $resposta[$i] ?> </option>
                <?php } ?>
```

```
</select><br>
<label for="poco">Poco : </label>
<select name="selpoco">
<?php for($j=0 ; $j<18; $j++ ){?>
<option value= <?php echo ($j+1) ?> >Poço <?php echo ($j+1) ?> </option>";
<?php } ?>
</select><br>
<input class="submit" name="it" type="submit" value="Visualizar"/><br>
<br><h1>Setar  Idletime</h1>
<label for="newidletime">Novo valor de IdleTime: </label>
<input type="text" name="newidletime" /><br>
<input class="submit" name="it" type="submit" value="Setar"/><br>
</form>
<?php break;
case "administrador":?>
…
}else{?>
        Você não tem permissão para acessar o sistema! <br>
…
```

Figura 3.5: Código PHP da página Idletime.

3.2.3 O Web Site

O site foi implementado com o intuito de prover uma interface amigável e prover um sistema de login seguro e que atenda a todas as funcionalidades que o Sistema de Monitoramento exige. Na página inicial, além de informações gerais de que trata esta página e de alguns links para páginas relacionadas como UFRN, Petrobras, ANP e DIMAP, existe um painel destinado a identificação do usuário para ter acesso aos serviços do Web Service. Neste painel o usuário entra com as informações de login, senha e perfil, e somente terá acesso ao sistema caso as três informações coincidam com os dados armazenados para cada usuário num banco de dados PostGreSQL. Os perfis de usuário são administrador, engenheiro, tecnico e estagiário. E suas funcionalidades podem ser vistas na tabela abaixo:

Perfil	Funcionalidades Disponíveis
Administrador	✓ Visualizar IdleTime ✓ Atualização dos dados dos campos ✓ Listar usuarios ✓ Cadastrar Usuários ✓ Descadastrar usuários ✓ Visualizar Estado de Bombeio ✓ Visualizar Carta Dinamométrica
Engenheiro	✓ Visualizar IdleTime ✓ Setar IdleTime ✓ Atualização dos dados dos Campos ✓ Visualizar Carta Dinamométrica ✓ Visualizar Carta Dinamométrica mais Atual ✓ Visualizar Estado de Bombeio ✓ Inverte Estado de Bombeio
Técnico	✓ Visualizar IdleTime ✓ Atualização dos dados dos campos ✓ Visualizar Estado de Bombeio ✓ Carta Dinamométrica
Estagiário	✓ Visualizar IdleTime ✓ Atualização dos dados dos campos ✓ Estado do Bombeio

Tabela 3.1—Funcionalidades de cada perfil de usuário.

Para esse controle de que usuário tem acesso a que tipo de informação foram utlizadas códigos em PHP com estruturas de switch (perfil) para tal controle. A página é montada dinamicamente exatamente no momento de sua abertura de acordo com o perfil de acesso do usuário. Um mapa do site pode ser mostrado abaixo na Figura 3.6:

Figura 3.6– Mapa do Site.

Deste mapa, deve-se frisar que na página chamada “Conectado“, aparece a estrutura de switch anteriormente citada, disponibilizando somente as opções pertinentes ao perfil de usuario conectado ao sistema. Nas páginas seguintes uma semelhante tomada de decisão ocorre. Por exemplo, na página “IdleTime“ quando o sistema deve diferenciar quando o usuário pode apenas visualizar ou pode ainda setar este valor.

3.2.4 Telas no Web Site

Uma vez planejado o mapa do site, as telas foram desenvolvidas com a ajuda da ferramenta Dreamweaver. Foram utilizados a tecnologia CSS (*Cascaded Style Sheets*) o que disponibiliza uma padronização entre as páginas com efeitos sutis de quadros, labels, select, botões e figuras, permitindo diferentes efeitos visuais na apresentação do site. Na Figura 3.7 pode ser visualizada a Página principal.

Esta página inicial é exibida quando se entra no sistema, e retorna-se a esta também, quando o usuário tentar acessar o sistema com um usuário inexistente, ou senha inválida, ou ainda um perfil de acesso diferente do registrado para este.

Quando o usuário entra com as informações corretas, a tela exibida mostra o nome *Conectado* no topo da página e na coluna ao lado direito, além da opção de Desconectar. É exibido um texto ao usuário mostrando seu login e seu perfil de acesso, além de uma saudação de bom dia, boa tarde ou boa noite, evidentemente referente ao horário que o usuário acessa o sistema. Esse horário é aquele da estação servidora do script PHP, podendo diferir do horário local onde o cliente está acessando, uma vez que este sistema permite que essas informações sejam dispostas de forma segura na Intranet.

Figura 3.7– Página Inicial.

São exibidos links para as opções de cada ação que o usuário pode tomar, opções essas que estão dispostas na Tabela 3.1 exibida anteriormente neste capítulo. A Figura 3.8 ilustra uma a tela exibida quando o usuário *hercules* obteve sucesso no login do sistema, ele tem perfil de *engenheiro*.

Nesta, o usuário tem a liberdade de escolher entre as funcionalidades que são atribuídas a seu perfil (associado ao cargo), como exemplo um engenheiro pode exibir o idletime ou o estado de bombeio de um poço específico, escolhendo o campo e o poço dentre as opções que estão sendo monitoradas, como mostra a Figura 3.9.

Como estas telas estão sendo exibidas num browser, é possível que o usuário abra estas opções em Novas Janelas, organizadas Lado a Lado, o que pode ser muito interessante para uma análise comparativa entre valores de *idle-time*, carta dinamométrica ou estado de bombeio.

Por fim na Figura 3.11 é exibida uma tela onde o administrador acaba de entrar no sistema e tem na sua lista de funcionalidades disponíveis, as opções de efetuar o cadastramento ou descadastramento de um usuário ao sistema.

Figura 3.8– Login aceito.

Figura 3.9– Requisição do Idle Time e do Estado de Bombeio.

Pode ainda exibir a carta dinamométrica, como mostra a Figura 3.10:

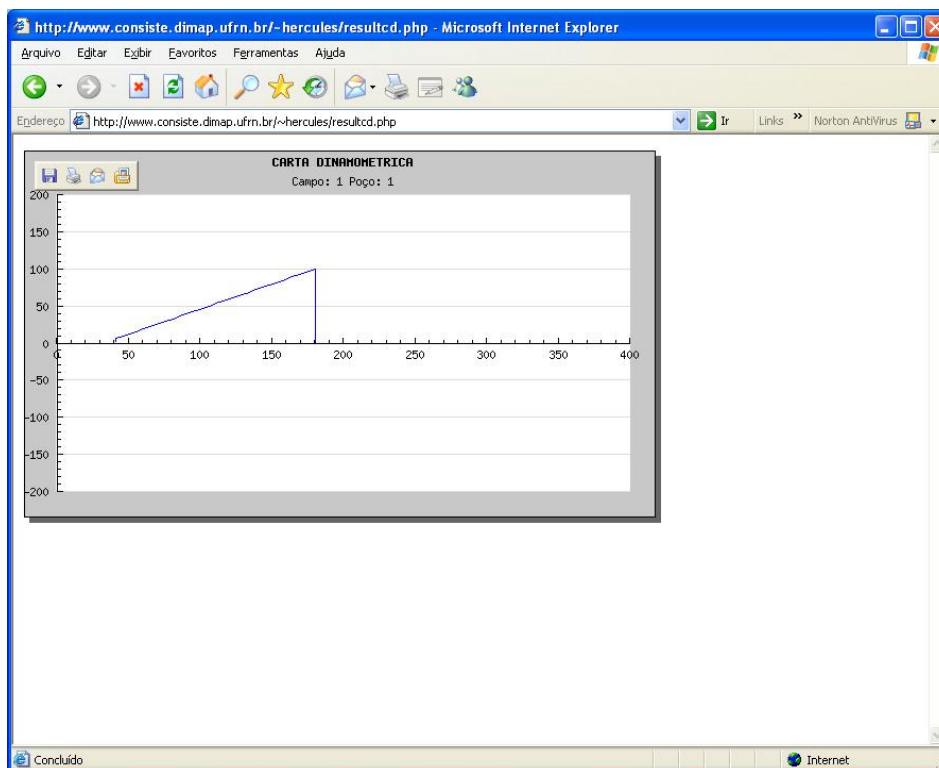



Figura 3.10– Resultado da Requisição da Carta Dinamométrica.

Figura 3. 11– Ações disponíveis ao Administrador.

3.2.5 Segurança

É importante comentar ainda que para o Controle de Acesso do usuário entre as páginas durante a navegação foi usada o sistema de sessões do script PHP.

Na programação para web é frequente a necessidade de manusear variáveis para que se tenha a percepção da sequencia de acesso do usuário. Antigamente qualquer transação comercial que realizada pela internet deveria ser feita sequencialmente do início ao fim em uma só visita. Para sanar esse problema existe duas estratégias o uso de cookies ou de sessions.

Criado em 1994 os cookies vieram ajudar a programação web, permitindo saber que páginas o usuário havia visitado anteriormente, a estratégia consiste simplesmente no fato de quando o usuário acessar um determinado site, este envia um pequeno arquivo à maquina do visitante, guardando os passos que este visitante já percorreu no site, ou ainda uma variável necessária de uma página para outra. Apesar de na criação do cookie existir a possibilidade de determinação de um tempo de expiração, este tempo influirá apenas na leitura ou não deste arquivo pelo site se o tempo ainda estiver válido, deixando o arquivo gravado em disco com informações muitas vezes sigilosas como login e senha.

Outra opção para garantir essa dinamicidade ao site é utilizar sessão (session), neste caso a sessão é criada no servidor e não mais no cliente. Quando o usuário entra no site é criada uma sessão e este recebe um numero identificador unico, session_id, válido enquanto a página estiver aberta, caso o usuário visite outro site ou feche o browser terá de efetuar o login novamente.

Por não gerar um arquivo em cada máquina visitante do site a session demonstra ser muito mais segura que os cookies. Outro ponto forte é que sessions não são válidas por um tempo expecífico como era com os cookies, então uma vez fora do site, se o usuário voltar, terá de efetuar o login novamente, mesmo que esteja fora por um curto espaço de tempo.

3.2.5.1 Cadastramento no banco de dados PostGreSQL

Ainda com relação ao quesito segurança, pode-se citar que o local onde estão armazenados login, senha e perfil do usuário, é em um banco de dados PostGreSQL, instalado num servidor diferente daquele onde está armazenado o Web site, protegida com senha de usuário e senha de acesso ao banco.

Esse banco está implementado de forma bem compacta com apenas uma tabela chamada *usuarios* contendo as principais informações a respeito dos usuários cadastrados. O codigo da criação e inserção inicial de usuários pode ser visto na Figura 3.12:

```
CREATE TABLE usuarios(
        nome VARCHAR(15) NOT NULL,
        senha VARCHAR(8) NOT NULL,
        perfil VARCHAR(15) NOT NULL,
        cpf CHAR(11),
        sexo BOOL DEFAULT 't',
        email TEXT NOT NULL,
        cidade VARCHAR(35),
        estado CHAR(2) DEFAULT 'RN',
        pais CHAR(3) DEFAULT 'BRA');
INSERT INTO usuarios (nome, senha, perfil, cpf, sexo, email, cidade, estado, pais) VALUES
('admin','teste01', 'administrador','23456789101','t','hercules@lcc.ufrn.br', 'Natal', 'RN', 'BRA');
INSERT INTO usuarios (nome, senha, perfil, cpf, sexo, email, cidade, estado, pais) VALUES
('hercules', 'teste02', 'engenheiro', '12345678910','t','hercules@engcomp.ufrn.br', 'Fortaleza', 'CE',
'BRA');
INSERT INTO usuarios (nome, senha, perfil, cpf, sexo, email, cidade, estado, pais) VALUES
('alysson', 'teste03', 'engenheiro', '01234567891','t','alysson@ppgsc.ufrn.br', 'Mossoro', 'RN', 'BRA');
INSERT INTO usuarios (nome, senha, perfil, cpf, sexo, email, cidade, estado, pais) VALUES ('thais',
        'teste04', 'engenheiro', '10123456789','f','thais@ufrnet.br', 'Joao Pessoa','PB', 'BRA');
INSERT INTO usuarios (nome, senha, perfil, cpf, sexo, email, cidade, estado, pais) VALUES ('pluto',
        'teste05', 'estagiario', '91012345678','t','pluto@lcc.ufrn.br', 'Recife', 'PE', 'BRA');
INSERT INTO usuarios (nome, senha, perfil, cpf, sexo, email, cidade, estado, pais) VALUES
('pateta', 'teste06', 'estagiario', '89101234567','t','pateta@lcc.ufrn.br', 'PontaPora', 'RN', 'BRA');
INSERT INTO usuarios (nome, senha, perfil, cpf, sexo, email, cidade, estado, pais) VALUES
('patonilo', 'teste07', 'estagiario', '78910123456','t','patolino@lcc.ufrn.br', 'Natal', 'RN', 'BRA');
INSERT INTO usuarios (nome, senha, perfil, cpf, sexo, email, cidade, estado, pais) VALUES
('pernalonga', 'teste08', 'tecnico', '67891012345','t','pernalonga@lcc.ufrn.br', 'Parauapebas','PA',
'BRA');
INSERT INTO usuarios (nome, senha, perfil, cpf, sexo, email, cidade, estado, pais) VALUES
('perninha', 'teste09', 'tecnico', '56789101234','t','perninha@lcc.ufrn.br', 'Pelotas','RS', 'BRA');
INSERT INTO usuarios (nome, senha, perfil, cpf, sexo, email, cidade, estado, pais) VALUES
('pluckduck', 'teste10', 'tecnico', '45678910123','t','pluck@lcc.ufrn.br', 'São Paulo', 'SP', 'BRA');
```

Figura 3. 12– Inserção de Usuários no Banco de Dados

No código acima temos somente uma tabela criada de nome *usuarios* com os campos *nome*, *senha*, *perfil*, *cpf*, *sexo*, *email*, *cidade*, *estado* e *país*. Sendo os três primeiros consultados para a autenticação do usuário e os outros importantes caso se queira saber como entrar em contato com este usuário.

4 Trabalhos Relacionados

Existem, no mercado, alguns Sistemas de Monitoramento e Controle de poços que serão brevemente apresentados nesse capitulo. Basicamente foram encontrados dois exemplos de aplicações voltadas para a área de exploração de poços de Petróleo.

O primeiro deles é o ePIC – Controlador de Bombeio Mecânico[13], trata-se de uma ferramenta voltada exclusivamente para o controle de Bombeio Mecânico. O segundo é o csLIFT[14], que é bem mais complexo que o primeiro e abrange várias outros tipos de extração de petróleo e gás.

4.1 ePIC – Controlador de Bombeio Mecânico

Este dispositivo microcontrolado, como pode ser visto na Figura 4.1, fica acoplado ao poço e efetua o controle inteligente de uma UBM através da leitura de sensores que vão acoplados ao controlador.

Figura 4.1– ePIC – Controlador de Bombeio Mecânico.

O ePIC é dotado de Inteligência e Controle na Locação do Poço, provendo os seguintes recursos avançados:

- Alarmes e diagnóstico do sistema, na locação do poço.
- Alarmes selecionados do sistema de bombeio baseado na análise das condições do poço, pelo computador.
- Alarmes de área da carta para detecção de haste partida e identificação de atrito de fundo.
- Linguagem Inteligente de Controle Programável (PICL - *Programmable Intelligent Control Language*) que fornece aos usuários flexibilidade em modificar o mecanismo de controle, adequando às condições específicas do poço.
- Controle das válvulas do poço e determinação do contrabalanceio da unidade, disponíveis nos sistemas de análises no computador.
- Melhoria no armazenamento de dados e tendências.
- Avançada função de aquisição de dados (*data-logging*) permitindo que qualquer registro seja adquirido e usado em aplicações lógicas.

O controlador, com um sistema patenteado de otimização de *idle time*, pode ser usado na otimização contínua do tempo do ciclo, mesmo na falha de energia, desgaste da bomba, vazamento na coluna ou "kicks" de produção. O controle de balanço de ar, balanceia automaticamente as unidades de bombeio balanceadas a ar. A opção do controle otimizado de picos de energia interrompe a operação do poço, durante os períodos de pico ou de alto custo de energia. O método de recuperação de falta de energia, permite aos poços bombear altos níveis de fluído, após falhas de energia. A função de prevenção de retorno da operação do motor elétrico pode ser usada, quando existir condensação presente nas bobinas do motor.

Quando ocorrer falhas no sistema de bombeio é possível programar o sistema para agir da maneira mais adequada, de acordo com a falha identificada, e tomar a iniciativa de

acionar um alarme, um luz de alerta, ou ainda, se necessário for, até parar o funcionamento do poço.

Uma desvantagem deste dispositivo é que não é possível acessá-lo remotamente, encontra-se acoplado direteamente a UBM e somente ali é permitido a visualização das variáveis monitoradas. Outro ponto que pode ser citado como desvantagem desse dispositivo é que este monitora e controla apenas um poço por vez, pois foi projetado para receber somente as informações de um poço e com base nessas realizar o processamento necessário para o controle do mesmo.

4.2 csLIFT –Pacote de Software para Otimização Total do Campo

O software csLIFT proporciona um sistema de automação integral, que incrementa a eficiência do bombeio, reduz falhas nos poços, e minimiza o *downtime*, que corresponde ao tempo em que o poço fica parado por motivo de falha. Este pacote Integra a engenharia de analises, o desenho com o monitoramento e o controle em tempo real dos poços e equipamentos em vazamentos de petróleo e gás. O pacote de software torna possível a otimização da produção ao reduzir o custo associado com a extração artificial e as outras operações de produção. csLIFT é um sistema cliente-servidor baseado em Windows, que trabalha com todos os principais controladores e CLP’s.

Os produtos inovadores de *Case Service* unem os sistemas de informação de uma companhia com dados em tempo real provenientes dos poços e demais equipamentos no campo. São pontos destacáveis:

- Re-coleção e controle remoto de dados.
- Suporte a equipamentos de múltiplos provedores.
- Gestão por exceções
- Gráficos de tendências históricas.
- Alarmes por exceção e inteligentes.

- Análises e historia de desempenho.
- Reportes preconfigurados ou personalizados, usando dados em tempo real e calculados.
- Fácil configuração online desde qualquer cliente.
- Rápida configuração de paging por alarmes.
- Arquitetura cliente / servidor.
- Acesso remoto através de acesso telefônico a redes.

O Case Service LIFT (csLIFT) é um pacote de software que tem por escopo atuar em todo e qualquer tipo de exploração possível de ser encontrada dentro de um campo de exploração de petróleo. Na Tabela 4.1 se podem conferir os diferentes softwares que contemplam este pacote e sua respectiva área de atuação:

Extração Artificial	
csBeam Suíte	Bombeio Mecânico
csSubs Suit	Bombas Eletro-Submergíveis
csPCP	Bombas de Cavidad Progressiva
csGasLift	Poços com Gás Lift
csPlungerLift	Poços com Plunger Lift
Operações de Produção	
csWellTest	Controle de Poços
csInjection	Injeção de Água/CO_2/Vapor
csEFM	Poços Gasíferos
csProduction	Produção
Tempo Real	
csMonitor	SCADA

Tabela 4.1 – Pacote csLIFT.

Por se tratar de um pacote que utiliza a arquitetura cliente/servidor é necessário instalar o software Servidor na máquina que servirá as outras máquinas. Em seguida deve ser instalado o software específico para o cliente em cada máquina cliente para que permita a instalação

O software csBeam, do pacote csLIFT, facilita supervisão controle e análise dos dados obtidos pelos controladores dos poços de bombeio mecânico. O sistema identifica os poços em situação de alarme, podem fazer a varredura dos controladores e exibir múltiplas cartas dinamometricas de vários poços ao mesmo tempo para que o usuário possa comparar resultados e fazer análises ou mesmo tomada de decisões. Alterações de parâmetros de produção podem ser impostas ao sistema e conferir-se o resultado diretamente nos valores de produção do poço.

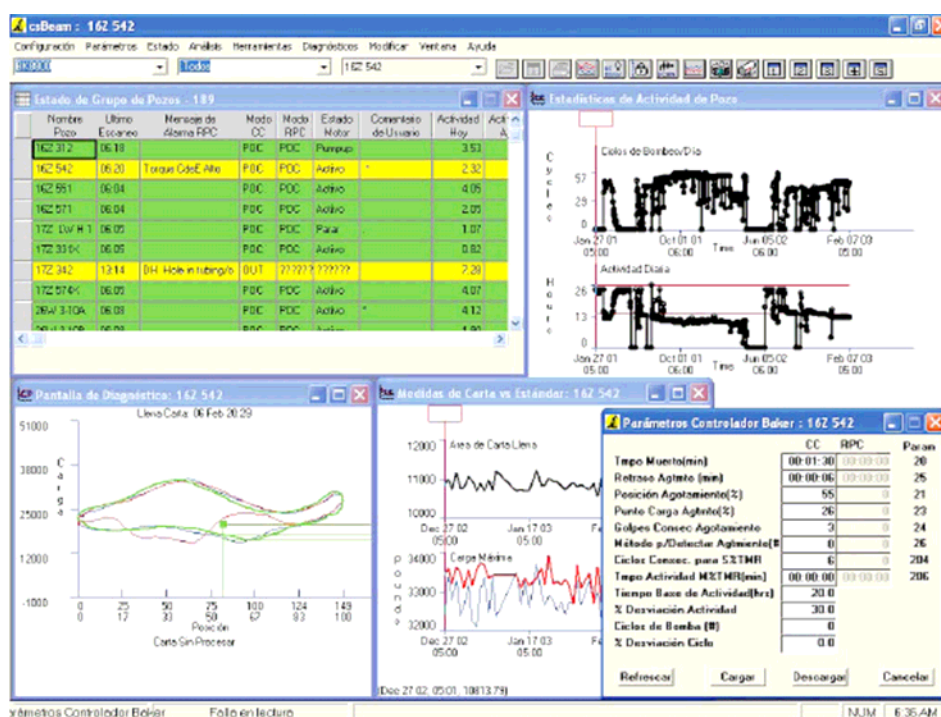



Figura 4.2 - csBeam Suíte.

Pelo fato de o trabalho em questão tratar apenas a exploração de poços explorados por Unidades de Bombeio Mecânico o mais semelhante deste é o csBream Suíte, Figura 4.2. Serão abordados apenas brevemente os outros Softwares deste pacote voltados à Extração Artifical cujo escopo abrange vários outras estratégias de exploração de petróleo e gás.

Figura 4.3 - csSubs Suit.

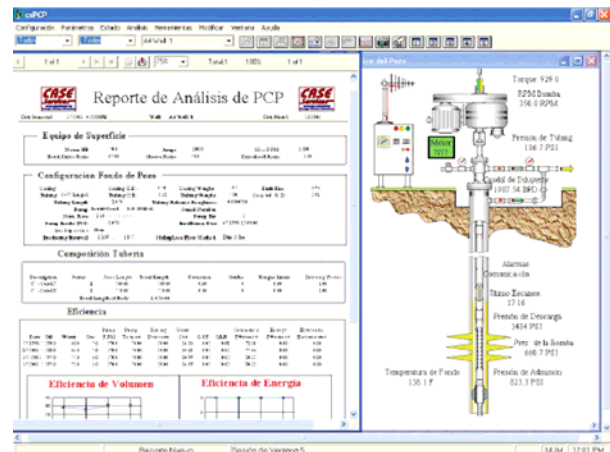



Figura 4.4 – csPCP.

Com o software csSubs Suit, Figura 4.3, é possível comparar por gráficos o desempenho da bomba e ao mesmo tempo valores de produção e a partir daí identificar desgaste da bomba, prever a vida útil remanescente da bomba e diagnosticar as causas de falha da bomba.

O software csPCP, Figura 4.4, tem a finalidade de melhorar a eficiência das bombas de cavidades coletivas, trabalhando tanto as bombas de velocidades fixas como as de velocidades variáveis. Cria curvas e tendências de velocidade (em RPM) do controlador, torque aplicado a bomba e inclusive detecta situações de pump-off da bomba.

O software csPlungerLIFT, Figura 4.5, monitora e cria curvas e tendências de velocidade do plunger, tensão na batería, número de chegadas do plunger, e outros parâmetros operacionais recoletados pelo controlador. Os gráficos se atualizam em tempo real, permitindo que o usuário identifique problemas potenciais e tome ações corretivas. Se o controlador tem a funcionalidade de medição de gas, csPlungerLift pode recolectar os dados de transferência de custodia. Uma cópia operacional de dados está disponível para sua edição em csPlungerLift, enquanto que uma versão sem editar é mantida para sua transferência a qualquer sistema de medição de gás.

O software csGasLift, Figura 4.6, proporciona análise e desenho para otimizar o desempenho de poços com gas lift. A configuração das válvulas e condições de produção

se usam para analisar o estado operativo de válvulas operadas por pressão ou de resposta proporcional. O software usa a informação do poço e correlações multifase para gerar o desenho ótimo.

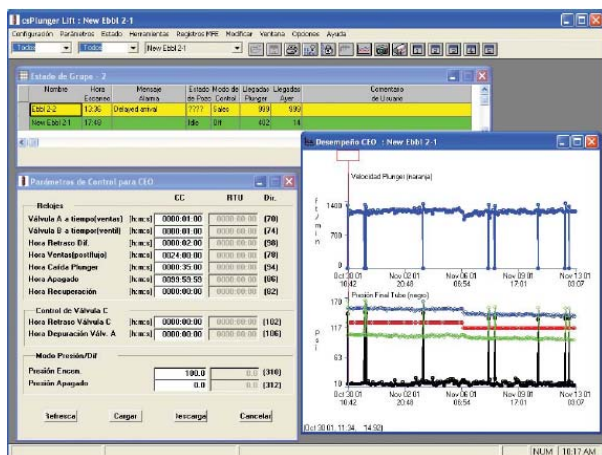



Figura 4.5 – csPlungerLift.

Figura 4.6 – csGasLift.

4.3 Conclusão sobre os Trabalhos Relacionados Analisados

Destes trabalhos relacionados pode-se concluir do e-PIC que apesar de dispor de uma ferramenta segura, confiável e inteligente, capaz de monitorar o poço e gerar cartas dinamométricas, não dispõe de outros recursos importantes como o acesso remoto. Já o csBeam Suite é uma ferramenta poderosa no apoio a extração de petróleo por Bombeio Mecânico, mostra Gráficos bastante elaborados e permite comparações entre eles, tem ainda um dispositivo interessante que armazena as cartas dinamométricas e variáveis coletadas. Mas mesmo o csBeam Suite que permite o acesso remoto, só o faz caso o usuário instale um software cliente específico para esse fim, pois sua arquitetura foi projetada para ser do tipo cliente/servidor. E ambos os trabalhos analisados não dispõem de um sistema de controle de acesso ao software, uma vez que qualquer usuário que tiver acesso à estação cliente poderá utilizar o sistema de monitoramento e ter acesso às informações.

5 Conclusão

Esse trabalho apresentou o desenvolvimento de um Sistema Web para monitoramento de poços de petróleo. O sistema consiste em um front-end para um mecanismo de monitoramento implementado na plataforma CORBA. Trata-se de um Web Service implementado em linguagem SOAP aplicado ao monitoramento de Poços de Petróleo em Campos de Exploração por Bombeio Mecânico.

A opção pela escolha de um site Web deve-se ao fato da facilidade de acesso a rede mundial de computadores mesmo nos lugares mais remotos. Dentre os recursos disponíveis se enconcontram a visualização de variáveis específicas de cada campo como idletime e o estado de bombeio além da informação gráfica da carta dinamométrica. É possível ainda interagir diretamente ligando ou desligando um poço e pedindo informações mais recentes.

O administrador tem controle sobre os usuários cadastrados permitindo a inclusão ou remoção de usuários ao sistema.

Para ter acesso ao sistema o usuário deve ter sido previamente cadastrado ao sistema, onde somente o administrador tem autorização para inserir e/ou remover usuários, bem como à lista de usuários.

O acesso ao Sistema ocorre depois de autenticado login, senha e perfil de acesso de maneira que somente terá acesso se as três informações requeridas estiverem corretas e conforme o cadastro do banco de dados.

5.1 Dificuldades Encontradas

As maiores dificuldades foram encontradas no desenvolvimento do Sistema Corba, uma vez que muito tempo foi investido para tentar inserir o Servidor Soap no Cliente Corba. Motivos como, pouca documentação do código e difícil comunicação com o desenvolvedor do código Legado, levaram a decisão de reimplementar o sistema CORBA na linguagem Java.

5.2 Contribuições

Este sistema pode trazer contribuições para monitoramento dos dados da exploração de petróleo. O fato de usuários situados remotamente terem acesso a dados específicos de cada poço em um campo de exploração terrestre, possibilita análises de produção, estratégias de exploração de uma bacia, estabelecimento de um programa de manutenção preventiva e corretiva dos equipamentos e uma diminuição dos custos operacionais de desenvolvimento.

Do ponto de vista de segurança, o poço poder ser desativado também remotamente evitando que o operador se exponha às condições de risco que este tipo de serviço oferece se realizado em campo. Ganha-se com isso redução de tempo e custo na execução deste procedimento.

5.3 Trabalhos Futuros

Testes exaustivos devem ser executados, para testar aspectos como escalabilidade e segurança. Os testes realizados foram restritos a 5 (cinco) acessos simultâneos ao sistema. Porém, para o fim a que se aplica tal ferramenta, mais testes devem ser executados neste aspecto.

Um outro trabalho futuro compreende um mecanismo para log dos valores monitorados. De posse de dados anteriores o usuário poderia fazer análises que o ajudem na tomada de decisões.

Outro fator de grande importância é que o sistema poderia implementar a opção de colocar lado a lado numa mesma página informações de dois poços distintos. Isso atualmente pode ser feito porém abrindo duas páginas de browsers diferentes.

O sistema também poderia ampliar o seu escopo de aplicação para outros tipos de exploração como Gás Lift, Bombeio Centrífugo Submerso, dentre outros. Não haveria grande complexidade em realizar essa tarefa pois vários aspectos do Bombeio Mecânico poderiam ser aproveitados.

Referências Bibliograficas

[1] THOMAS, J.E., Fundamentos de Engenharia de Petróleo, Editora Interciência, Rio de Janeiro, 2001.

[2] SALIM, Diogo. "Sistema Distruído para Monitoramento de Poços de Petróleo com Elevação Artificial". Graduação em Engenharia de Computação, UFRN, 2004.

[3] VARGAS, Patrícia Kayser. CORBA: Visão Geral e Aspectos Relacionados à Replicação e Persistência. Disponível em: <http://www.inf.ufrgs.br/~kayser/cmp157/T1/T1Corba.html>. e RAVAGNANI, Wanderley Júnior. Arquitetura CORBA. Disponível em: <http://www.ic.unicamp.br/~ra007293/corba/corba.html>.

[4] http://www.w3.org/TR/soap/

[5] Relatório Produção Nacional de Petróleo e LGN (barris), <http://www.anp.gov.br/doc/dados_estatisticos/Producao_de_Petroleo_b.xls>

[6] POTTS, S.; KOPACK, M..Aprenda Web Services em 24 horas, Editora Campus, Rio de Janeiro, 2003

[7] I-WEB, “Web Services“ Outubro2003 <http://www.iweb.com.br/iweb/pdfs/20031008-webservices-01.pdf>

[8] http://www.w3.org/Protocols/

[9] CLEMBET, K.A.S.,Introdução ao PHP e PostGreSQL,2003, <http://www.lcc.ufrn.br/~clembet/download/introPHP.zip>

[10] The PostgreSQL Global Development Group, Tutorial do PostgreSQL 7.3.2 http://www.postgresql.org.br/downloads/tutorial.pdf

[11] Extending Dreamweaver, <http://download.macromedia.com/pub/documentation/en/dreamweaver/mx2004/extending_dw.pdf>

[12] SourceForge.net, Project: NuSOAP - SOAP Toolkit for PHP <http://sourceforge.net/projects/nusoap/>

[13] ePIC – Controlador de Bombeio Mecânico, <*http://www.ep-agent.com/PDFs/eP_L/L_ePIC_Rod_Pump_Controller_Portuguese.pdf*>

[14] csLIFT Automation/Analysis Software <http://www.ep-solutions.com/Solutions/Case/Automation_Analysis_Software.htm>